基于差分进化的复杂软件系统动态可靠性分配 *

张世金 [1]，张国富 [1, 2, 3]，苏兆品 [1, 2, 3]，岳　峰 [1, 2, 3]

(1. 合肥工业大学 计算机与信息学院，合肥 230601; 2. 工业安全与应急技术安徽省重点实验室，合肥 230601; 3. 安全关键工业测控技术教育部工程研究中心，合肥 230601)

摘　要：现有关于复杂软件系统可靠性分配的研究均基于结构固定的软件系统，而实际情况中软件系统结构往往不固定。针对这一矛盾，构建复杂软件系统动态可靠性分配优化模型，并基于差分进化设计复杂软件系统动态可靠性分配算法。在系统结构发生变化时，首先基于 D-S 证据理论对系统中各模块的全局权重重新进行评估，并考虑变化前后系统的关联性，在差分进化生成初始种群时保留了部分历史解。最后，通过仿真实验分析验证了所提方法的有效性。

关键词：复杂软件系统；动态可靠性分配；差分进化；D-S 证据理论；保留历史解

中图分类号：TP311.5　　doi: 10.3969/j.issn.1001-3695.2018.03.0265

Dynamic reliability allocation of complex software system based on differential evolution

Zahng Shijin[1], Zahng Guofu[1, 2, 3], Su Zhaopin[1, 2, 3], Yue Feng[1, 2, 3]

(1. *School of Computer & Information*, *Hefei University of Technology*, *Hefei* 230601, *China*; 2. *Anhui Province Key Laboratory of Industry Safety & Emergency Technology*, *Hefei* 230601, *China*; 3. *Engineering Research Center of Safety Critical Industrial Measurement & Control Technology for Ministry of Education*, *Hefei* 230601, *China*)

Abstract: Existing research on the reliability allocation of complex software system just based on software systems whose structure were fixed, but the structure of software systems always change in practice. For this contradiction, this paper presents a dynamic reliability allocation optimization model of complex software system. At the same time, developing a dynamic reliability allocation algorithm of complex software system based on differential evolution. Specifically, when the system structure changes, we first estimate the parameters of each module according to the Dempster-Shafer evidence theory. Next, taking account of the relevance of the system before and after the change, we retain some historical solutions in the previous search to form part of the initial population in differential evolution. Finally, verifying the effectiveness of the proposed approach by simulation experiments.

Key words: complex software system; dynamic reliability allocation; differential evolution; Dempster-Shafer envidence theory; reserve historical solutions

0　引言

可靠性设计一直是软件工程领域一个非常重要和活跃的方向。随着大规模复杂软件系统不断在电力系统、铁路运输、航空航天、国家安全等安全关键领域内的广泛应用，对软件系统的可靠性的要求越来越高，软件可靠性分配问题也越来越受到研究者和设计者的关注[1~3]。但是，软件可靠性无法精确测量，只能通过对软件系统进行测试来度量其可靠性，对于软件开发工程师来说，为达到最优的可靠性，在开发的早期阶段完成软件可靠性分配是软件工程的一个重要环节[3]。

在可靠性分配问题建模方面，Zahedi 等人[4]以成本为约束构建软件系统可靠性最大化模型。Leung[5, 6]根据操作剖面定义软件可靠性函数，并考虑到每个客户的软件操作剖面，同时满足成本预算和软件质量要求等实际约束条件。Kapur 等人[7]将系统冗余与预算约束相结合，使系统可靠性最大化，而且还考虑了可供不同模块使用的替代品之间的兼容性问题。Roy 等人[8]针对多功能多用户数字中继系统设计了一种软件可靠性分配方案以提高输电线路故障的检测、分类和定位的可靠性。Chatterjee 等人[9]提出了一种结合用户视点的软件可靠性分配模型，为了将用户需求和偏好纳入软件的技术设计和可靠性，

收稿日期：2018-03-28；**修回日期：**2018-05-22　　**基金项目：**国家自然科学基金项目(61573125, 61371155)；安徽省自然科学基金资助项目(1608085MF131, 1508085MF132，1508085QF129)；中央高校基本科研业务费专项资金资助项目（JZ2017YYPY0232）

作者简介：张世金（1991-），男，安徽怀宁人，硕士研究生，主要研究方向为基于搜索的软件工程（zsjtyut@163.com）；张国富（1979-），男，安徽合肥人，副教授，博士，主要研究方向为联盟博弈、进化计算、软件工程等；苏兆品（1983-），女，山东菏泽人，副教授，博士，主要研究方向为复杂智能系统；岳峰（1981-），男，副研究员，博士.

建立了一个系统层次结构，将用户的系统视图与软件管理人员和程序员结合起来。

在可靠性分配问题求解方面，韩冰青等人[10]基于模拟退火遗传算法，将软件系统的可靠性分配问题表达为带约束条件的组合优化问题，并采用模拟退火的适应度拉伸方法和多点交叉操来提高解的质量。徐悦等人[11]基于罚函数的混合分布估计和自适应交叉差分进化[12]（differential evolution, DE）的优化算法对顺序、并发、循环、容错四种体系结构的复杂软件可靠性进行评估。Sangeetha 等人[13]利用多目标遗传算法求解考虑可靠性、成本和进度的多目标可靠性分配问题。然而，上述已有工作只是针对单一软件系统结构，在一些复杂软件系统中往往包含多个软件。而且，上述工作大都采用层次分析法预测各模块的参数值，没有考虑实际工程中的不确定性和不完全性。为此，Yue 等人[14]基于复杂软件系统层次结构，构建成本约束的系统可靠性最大化模型，并利用 D-S 证据理论[15]估计各模块的参数值，并基于差分进化搜索最优可靠性分配方案。

不过，上述已有工作都是考虑静态场景下的软件可靠性分配问题。需要注意的是，在实际的软件开发过程中，由于用户需求的变化或者开发人员的程序改进，导致软件系统的结构往往并不固定，变化之前建立的可靠性分配方案显然不能再适应变化后的系统结构。而根据现有的结构完全重新优化新的可靠性分配方案又显得费时费力。针对这一情景，本文首先提出一种针对复杂软件系统的动态可靠性分配模型，模拟复杂软件系统结构会发生变化的可靠性分配问题，并基于 D-S 证据理论和 DE 算法求解这一动态优化问题，同时在优化时考虑历史解以提高最终解的质量，最后通过仿真实验验证了所提方法的有效性。

1　问题描述

通常，复杂软件系统层次结构[14]是自上而下进行的，如图 1 所示。

第一层为软件层，是用户对每个软件 S_l 可靠性 $R_l(l=1,\cdots,p)$ 的总体评估。第二层为功能层，$F_k(k=1,\cdots,f)$，表示每一软件都存在若干功能。第三层为程序层，$P_i(i=1,\cdots,n)$，表示每一功能都由若干程序实现。第四层为模块层，$M_j(j=1,\cdots,m)$，表示每一程序都包含若干模块这里每个模块是独立的，且只将可靠性指标分配到模块层为止。

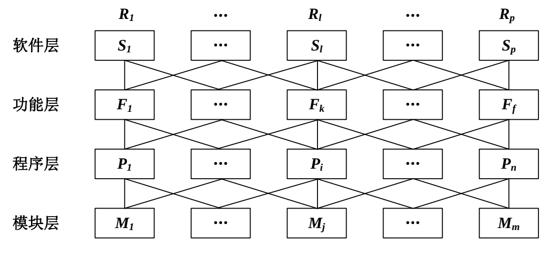


图 1　复杂软件系统层次结构

通常，用户在软件中执行各种功能的可靠性程度通过“软件实用性”来衡量。在上述层次结构中，程序层既反映了用户对软件可靠性的观点，又反映了程序员的建议，因此，通常从程序层的视角去研究复杂软件系统的实用性 U 。

$$U=\sum_{i=1}^{n} w_{P_i} r_{P_i} \tag{1}$$

其中：w_{P_i} 为程序 P_i 的全局权重，r_{P_i} 为程序 P_i 的可靠性。由于模块是独立的单元，所以 r_{P_i} 可根据下式计算：

$$r_{P_i}=\prod_{M_j\in P_i} r_{M_j} \tag{2}$$

其中：r_{M_j} 为模块 M_j 分配到的可靠性。此外，每个模块 M_j 有一个全局权重 w_{M_j} 。

复杂软件系统动态可靠性分配问题即是在一定的预算约束下，完成复杂软件系统中各模块的可靠性分配，使得系统的实用性达到最大。

$$\max U=\sum_{i=1}^{n}\left(w_{P_i}\prod_{j=1\wedge M_j\in P_i}^{m} r_{M_j}\right) \tag{3}$$

s.t.

$$0\le r_{M_j}\le 1,\ j=1,\cdots,m \tag{4}$$

$$a_{M_j}+b_{M_j}r_{M_j}\le w_{M_j}\sum_{l=1}^{p}B_{S_l},\ j=1,\cdots,m \tag{5}$$

$$\sum_{j=1\wedge M_j\in S_l}^{m}\left(a_{M_j}+b_{M_j}r_{M_j}\right)\le B_{S_l},\ l=1,\cdots,p \tag{6}$$

在上述模型中，m 的值是动态变化的，即模块数是变化的。目标函数 U 为整个复杂软件系统的整体实用性，是各模块可靠性 r_{M_j} 的函数。其中，r_{M_j} 是唯一的未知变量，因为 m 的值是动态变化的，所以变量集 $r_{M_1},\cdots,r_{M_j},\cdots,r_{M_m}$ 也是动态变化的。约束条件式(4)是对模块 M_j 可靠性的约束；约束条件式(5)是对模块 M_j 成本的预算约束，其中 a_{M_j} 为模块 M_j 达到可靠性 r_{M_j} 的固定开销成本，b_{M_j} 为可变成本，即 M_j 提高一个单位的可靠性所要增加的成本，B_{S_l} 为软件 S_l 的预算成本，$w_{M_j}\sum_{l=1}^{p}B_{S_l}$ 为模块 M_j 的最大预算。约束条件式(6)是对软件 S_l 的成本约束。

2　动态可靠性分配算法

2.1　差分进化

DE 是一种基于群体的启发式搜索算法[12]。主要的进化流程包括变异、交叉和选择。

DE 算法通过差分策略实现个体变异，常见的差分策略是随机选取种群中两个不同的个体，将其向量差缩放后与原始个体 X 进行向量合成生成一个新的变异个体 V 。

$$V=X+F\left(X_{r_1}-X_{r_2}\right) \tag{7}$$

其中：X_{r_1}、X_{r_2} 是两个随机选取的与 X 不相同的个体，$F\in(0,2)$ 为缩放因子。

交叉操作的目的是从原始个体 X 和变异个体 V 中根据交叉概率 CR 交换基因值产生新的试验个体 U 。如果随机数 $rand(0,1)\le CR$，$U=V$；否则 $U=X$ 。再通过适应度函数对试验

个体 U 进行评价，选择较优的个体组成下一代新种群。如果 U 的适应度值优于 X，则 $X=V$；否则 X 保持不变。

2.2 D-S 证据理论

D-S 证据理论[15]通过合并多重证据从而做出决策，对推理进行合理的解释而达到一定程度的信念。Yue 等[14]基于 D-S 证据理论来估计每个程序和模块的全局权重。在本文中，我们仍然采用这种方法对变化后的系统结构中各个程序和模块的全局权重重新进行估计。

设 U_X 是变量 X 可能值的样本空间，则称 U_X 为 X 的一个识别框架。X 的基本概率分布函数 U 满足：$\mu: 2^{U_X} \to [0,1]$；$\mu(\varnothing)=0$；$\sum_{A\in 2^{U_X}} \mu(A)=1$。其中，$2^{U_X}$ 为 U_X 的所有非空子集的集合，$\mu(A)$ 表示对 A 的精确信任度，为集合 A 的基本概率数。当来自不同数据源的数据对相同的识别框架提供不同的评估信息时，可用如下的合成规则被用于信息融合。

设 $\mu_1(B)$ 和 $\mu_2(C)$ 为同一识别框架的两个不同的证据，则

$$\mu_{12}(A)=\begin{cases}0, A=\varnothing\\ \sum\limits_{B\cap C}\mu_1(B)\cdot\dfrac{\mu_2(C)}{(1-K)}, A\neq\varnothing\end{cases} \tag{8}$$

$$K=\sum_{B\cap C=\varnothing}\mu_1(B)\cdot\mu_2(C) \tag{9}$$

其中：$A=B\cap C$；K 为归一化常数，用来计算所有非空集合的基本概率之和，可以避免在组合时将非零的概率赋给空集。通过 K 可以把空集所丢弃的信度按比例分配到非空集上，有效的表示了证据间的冲突程度，其值越大说明证据之间的冲突越大。

在本文中，首先根据各个软件的预算 B_{S_i} 确定各个软件的全局权重 $w_{S_i}=B_{S_i}\Big/\sum_{l=1}^{p}B_{S_l}$；然后让用户和程序员根据每个功能、程序、模块在各自所属的软件、功能、程序中的重要程度，给出每个功能、程序、模块在各自的软件、功能、程序中的一个局部权重，即图 1 中的每条线上都有一个局部权重值。然后根据文献[14]的计算和合成方法给出每个程序、模块在整个复杂软件系统中的全局权重 w_{P_i} 和 w_{M_j}。值得注意的是，当模块数发生变化时，只需要根据先前的每个程序的全局权重（并没有发生变化）、每个模块新的局部权重（发生了变化）来估计每个模块的全局权重。

2.3 算法描述

如图 2 所示，在优化过程中，如果系统结构发生变化，首先利用 D-S 证据理论基于每个模块新的局部权重估计各模块的全局权重，然后以式(3)为适应度函数，以式(4)~(6)为约束条件，基于 DE 算法搜索新系统结构下的最优可靠性分配方案。特别地，在初始化种群时，根据个体的适应度值排序，保留历史搜索中最好的 75%的个体作为当前优化的部分初始解，其余 25%的个体仍然根据约束条件随机初始化。

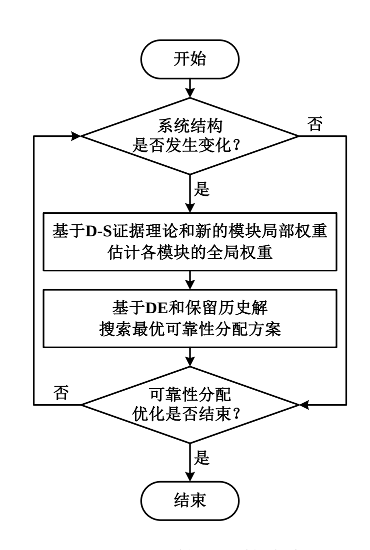
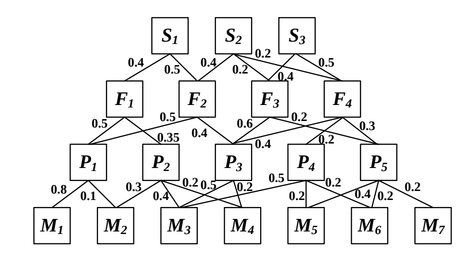


图 2　动态可靠性分配算法流程图

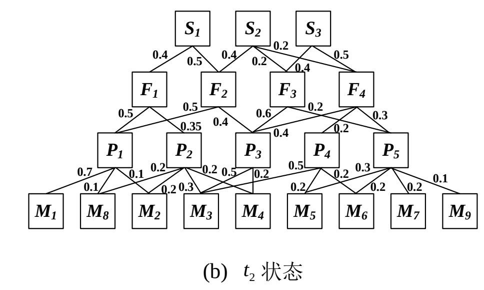
3 仿真实验与分析

3.1 实验环境

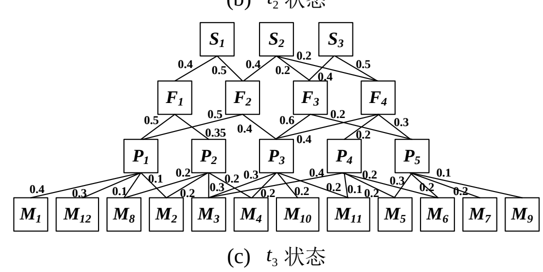
为了验证本文方法的有效性，在优化过程中设计了一个复杂软件系统，该系统包含 t_1、t_2 和 t_3 三个状态。三种状态下的系统层次结构和各个单元的局部权重如图 3 所示。



(a)　t_1 状态



(b)　t_2 状态



(c)　t_3 状态

图 3　各状态下的系统层次结构和局部权重

S_1、S_2 和 S_3 的预算分别为 250、450 和 300。状态 t_2 相比于状态 t_1 的变化在于程序 P_1 与 P_2 新增了一个模块 M_8，程序 P_5 新增了一个模块 M_9。相应的 P_1、P_2 和 P_5 关联的模块的权重也发生了改变，而 P_3 和 P_4 包含的模块的局部权重没有变化。状态 t_3 相比于状态 t_2 的变化在于程序 P_1 新增了一个模块 M_{12}，程序 P_3 新增了两个模块 M_{10} 和 M_{11}，程序 P_4 新增了一个模块 M_{11}。相应的 P_1、

P_3 和 P_4 关联的模块的局部权重发生了改变，而 P_2 和 P_5 包含的模块的局部权重没有变化。

此外，在 DE 算法中，交叉概率 $CR = 0.5$，缩放因子 $F = 0.94$，种群规模为 60。

3.2 实验结果

基于 D-S 证据理论得到程序层的全局权重为

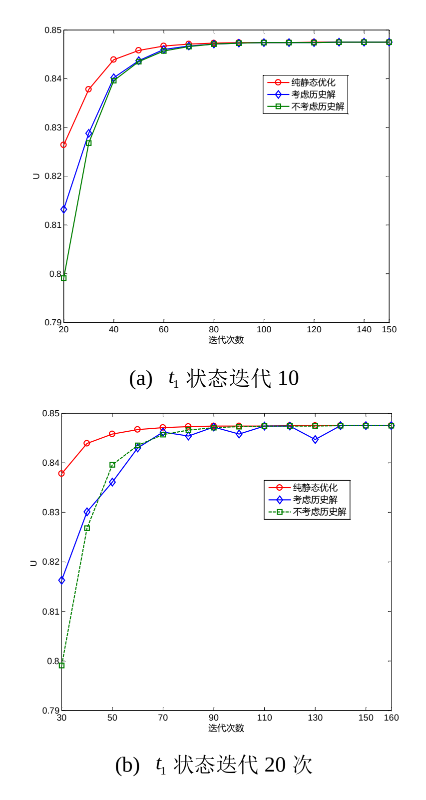
$$w_{P_i} = \{0.2628, 0.0284, 0.5106, 0.0582, 0.14\}$$

三种状态下模块层的全局权重如表 1 所示。

表 1　各状态下的模块层的全局权重

状态	w_{M_j}
t_1	{0.2239,0.0353,0.4007,0.2165,0.0619, 0.0359,0.0256}
t_2	{ 0.1962,0.0329,0.3977,0.2168, 0.049,0.036,0.0257,0.0329,0.0128}
t_3	{0.0997,0.0293,0.2364,0.1418,0.0436,0.032, 0.0228,0.0293,0.0114,0.1421,0.1368,0.0748}

图 4 给出了 DE 在状态 t_1 下分别迭代 10 次、20 次和 30 次后系统结构变为状态 t_2 后的优化结果。其中，“纯静态优化”指的是 DE 直接在状态 t_2 下搜索；“考虑历史解”指的是在系统结构变为状态 t_2 后，DE 将 t_1 状态下的部分历史解作为初始种群来搜索；“不考虑历史解”指的是在系统结构变为状态 t_2 后，DE 直接随机生成初始种群来搜索。“考虑历史解”和“不考虑历史解”均是属于动态优化。从图 4 可以看出，考虑历史解要比不考虑历史解明显收敛的快，说明保留前一状态下的历史解能够缩短初始种群与最优种群的距离。而且，还可以看出，在状态 t_1 迭代次数较少的情形下，动态优化也可以找到和纯静态优化相似的解，这是因为状态 t_2 下有足够的迭代次数以保证 DE 的充分探索。



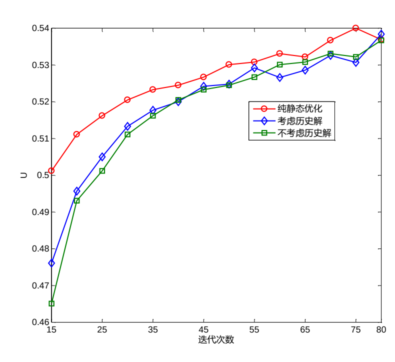
(a)　t_1 状态迭代 10

(b)　t_1 状态迭代 20 次

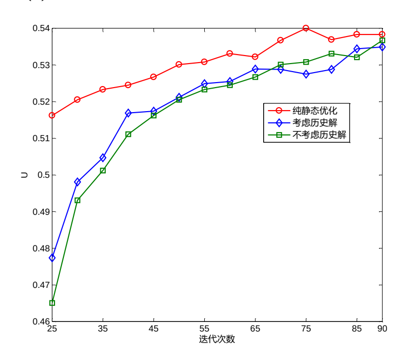
(c)　t_1 状态迭代 30 次

图 4　DE 从状态 t_1 到状态 t_2 后的优化结果

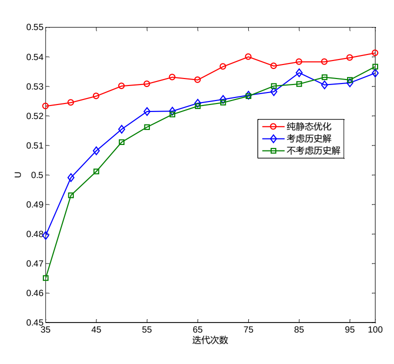
图 5 给出了 DE 在状态 t_1 下迭代 5 次后状态 t_2 下迭代 5 次、状态 t_1 下迭代 10 次后状态 t_2 下迭代 10 次和状态 t_1 下迭代 15 次后状态 t_2 下迭代 15 次后系统结构变为状态 t_3 后的优化结果。其中，“纯静态优化”指的是 DE 直接在状态 t_3 下搜索；“考虑历史解”指的是在系统结构变化后，DE 将前一状态下的部分历史解作为当前状态下的初始种群来搜索；“不考虑历史解”指的是在系统结构发生变化后，直接在当前状态下随机生成初始种群来搜索。从图 5 可以看出，随着变化频率的增加，动态优化和静态优化的差距越来越明显，说明系统结构变化频繁不利于可靠性分配问题的优化。但是可以看出，考虑历史解仍然要优于不考虑历史解。而且，在总的迭代次数较少、前面状态下迭代次数较大的情况下，考虑历史解的优势尤其明显，因为最后一个状态没有足够的迭代次数来保证 DE 的充分探索。因此，在恶劣情形下的动态优化，考虑历史解是一个提高最终解质量的一个有效手段。



(a)　t_1 状态迭代 5 次后 t_2 状态迭代 5 次



(b)　t_1 状态迭代 10 次后 t_2 状态迭代 10 次



(c) t_1 状态迭代 15 次后 t_2 状态迭代 15 次

图 5 DE 从状态 t_1 到状态 t_2 再到状态 t_3 后的优化结果

4 结束语

针对用户需求的变化或程序员的改进，在可靠性分配优化过程中复杂软件系统的结构可能会随着时间发生变化，本文构建了复杂软件系统动态可靠性分配优化模型，并基于 D-S 证据理论和差分进化设计了复杂软件系统动态可靠性分配算法。在系统结构发生变化时，首先基于 D-S 证据理论对系统中各模块的局部权重进行估计，并在差分进化生成初始种群时保留部分历史解。仿真实验结果验证了所提方法的有效性，但需要注意的是当模块数增加至比较多时可能导致问题的目标函数无解，第二组实验的结果也表明当模块数增加至一定数目时，目标函数的解空间会变得非常小，这会极大的限制差分进化的性能，因此本文提出的算法仅适于模块数较少时的情形。在后续的工作中，需要考虑：动态多目标可靠性分配问题；当软件层、功能层或程序层发生变化时的优化问题；根据模型推演 r_{M_j} 的上下界，设计约束处理技术以提高算法的搜索性能。

参考文献：

[1] 徐仁佐，张良平，张大帅. 软件可靠性分配的一个非线性规划模型 [J]. 计算机工程, 2003, 29 (17): 34-36. (Xu Renzuo, Zhang Liangpin, Zhang Dashuai. A nonlinear programming model of software reliability allocation [J]. Computer Engineering, 2003, 29 (17): 34: 36.)

[2] 高峰，仵林博，岳旸，等. 基于 AHP 与 AdaBoosting 的软件可靠性组合模型 [J]. 计算机工程, 2017, 43 (12): 69-72. (Gao feng, Wu Lingbo, Yue Yang, *et al.* Software reliability combination model based on AHP and AdaBoosting [J]. Computer Engineering, 2017, 43 (12): 69-72.)

[3] 张策，孟凡超，考永贵，等. 软件可靠性增长模型研究综述 [J]. 软件学报, 2017, 28 (9): 2402-2430. (Zhang Ce, Meng Fanchao, Kao Yonggui, *et al.* Survey of software reliability growth model [J]. Journal of Software, 2017, 28 (9): 2402-2430.)

[4] Zahedi F, Ashrafi N. Software reliability allocation based on structure, utility, price, and cost [J]. IEEE Trans on Software Engineering, 1991, 17 (4): 345-356.

[5] Leung Y W. Optimal reliability allocation for modular software system designed for multiple customers [J]. IEICE Trans on Information & Systems, 1996, 79 (12): 1655-1662.

[6] Leung Y W. Software reliability allocation under an uncertain operational profile [J]. Journal of the Operational Research Society, 1997, 48 (4): 401-411.

[7] Kapur P K, Bardhan A K, Jha P C. Optimal Reliability Allocation Problem for a Modular Software System [J]. Opsearch, 2003, 40 (2): 138-148.

[8] Roy D S, Mohanta D K, Panda A K. Software reliability allocation of digital relay for transmission line protection using a combined system hierarchy and fault tree approach [J]. IET Software, 2008, 2 (5): 437-445.

[9] Chatterjee S, Singh J B, Roy A. A structure-based software reliability allocation using fuzzy analytic hierarchy process [J]. International Journal of Systems Science, 2015, 46 (3): 513-525.

[10] 韩冰青，高建华. 基于模拟退火遗传算法的软件可靠性分配及研究 [J]. 计算机工程, 2003, 29 (4): 67-69. (Han Bingqing, Gao Jianhua. Reliability allocation and research of software system based on Simulated Annealing Genetic Algorithm [J]. Computer Engineering, 2003, 29 (4): 67-69.)

[11] 徐悦，皮德常. 基于混合智能优化算法的复杂软件可靠性分配 [J]. 软件学报，2018，29 (9): 1-19. (Xu Yue, Pi Dechang. Complex software reliability allocation based on hybrid intelligent optimization algorithm [J]. Journal of Software, 2018, 29 (9): 1-19.)

[12] Das S, Mullick S S, Suganthan P N. Recent advances in differential evolution: an updated survey [J]. Swarm & Evolutionary Computation, 2016, 27: 1-30.

[13] Sangeetha M, Arumugam C, Kumar K M S, *et al.* An effective approach to support multi-objective optimization in software reliability allocation for improving quality [J]. Procedia Computer Science, 2015, 47: 118-127.

[14] Yue F, Zhang G, Su Z, *et al.* Multi-software reliability allocation in multimedia systems with budget constraints using Dempster–Shafer theory and improved differential evolution [J]. Neurocomputing, 2015, 169: 13-22.

[15] Yager R R, Liu L. Classic Works of the Dempster-Shafer Theory of Belief Functions [M]// Classic Works on the Dempster-Shafer Theory of Belief Functions (Studies in Fuzziness and Soft Computing) . New York: Springer-Verlag, 2007: 1-34.